IBM Compilers for pSeries

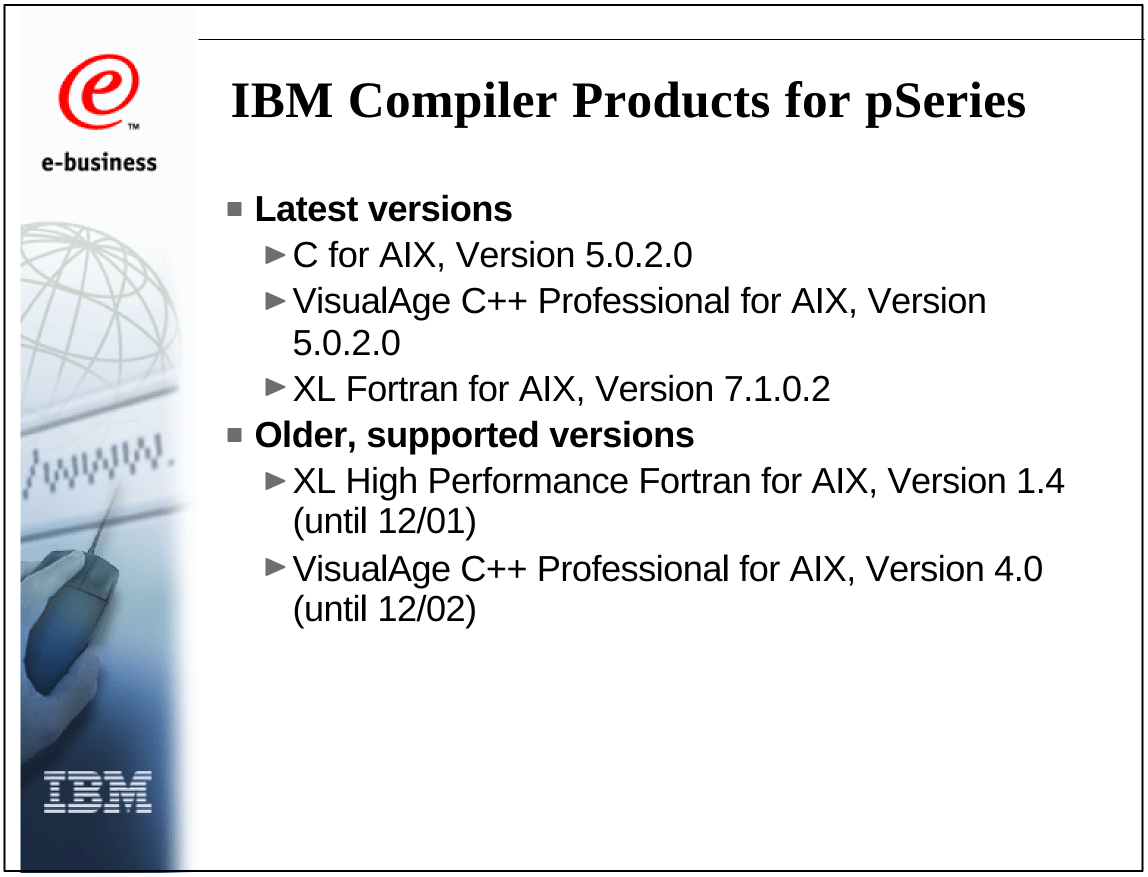
SCICOMP

IBM SP Scientific Computing User Group

October 10, 2001
Bob Blainey
blainey@ca.ibm.com

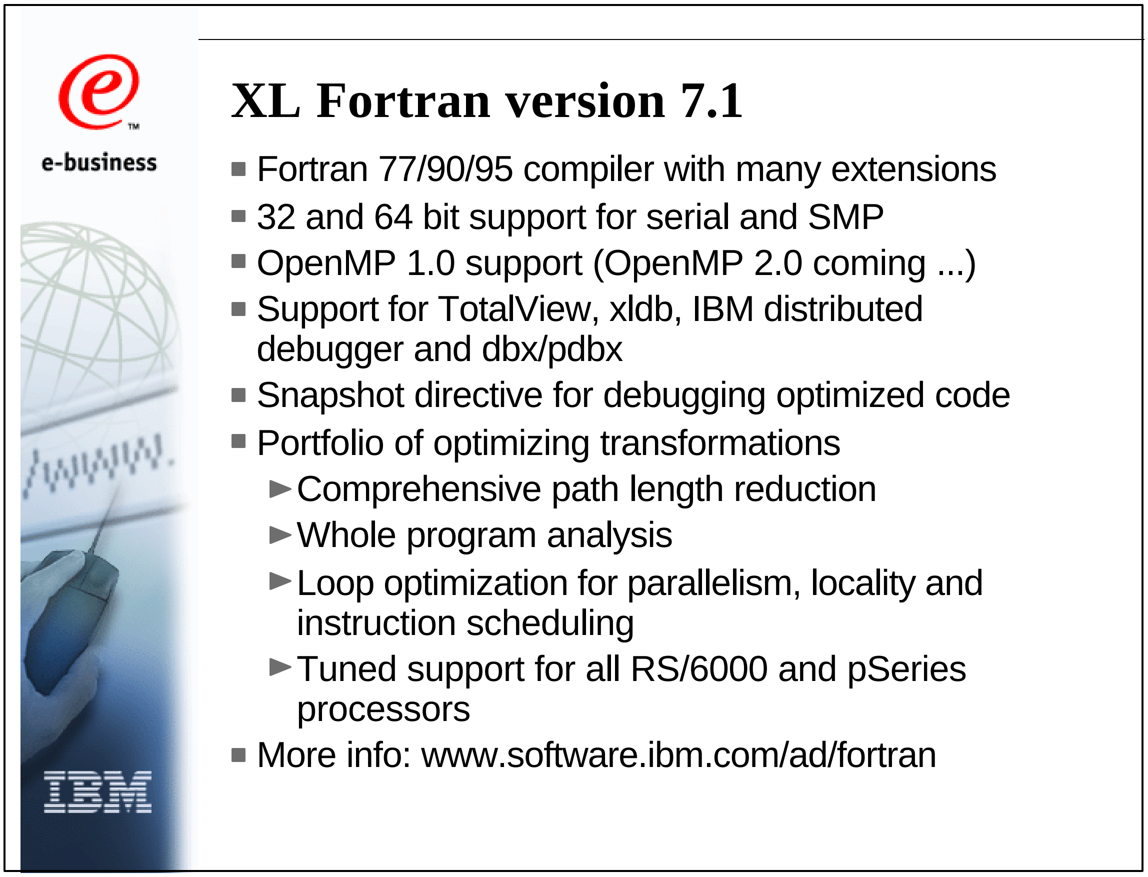
Agenda

- Review of the pSeries compiler products
- Compiler Configuration Options
- Compiler roadmap
- Some Performance Results
- A short tutorial on performance controls
- A peek inside the compiler
- Q&A



IBM Compiler Products for pSeries

- **Latest versions**
  - C for AIX, Version 5.0.2.0
  - VisualAge C++ Professional for AIX, Version 5.0.2.0
  - XL Fortran for AIX, Version 7.1.0.2
- **Older, supported versions**
  - XL High Performance Fortran for AIX, Version 1.4 (until 12/01)
  - VisualAge C++ Professional for AIX, Version 4.0 (until 12/02)



XL Fortran version 7.1

- Fortran 77/90/95 compiler with many extensions
- 32 and 64 bit support for serial and SMP
- OpenMP 1.0 support (OpenMP 2.0 coming ...)
- Support for TotalView, xldb, IBM distributed debugger and dbx/pdbx
- Snapshot directive for debugging optimized code
- Portfolio of optimizing transformations
  - Comprehensive path length reduction
  - Whole program analysis
  - Loop optimization for parallelism, locality and instruction scheduling
  - Tuned support for all RS/6000 and pSeries processors
- More info: www.software.ibm.com/ad/fortran

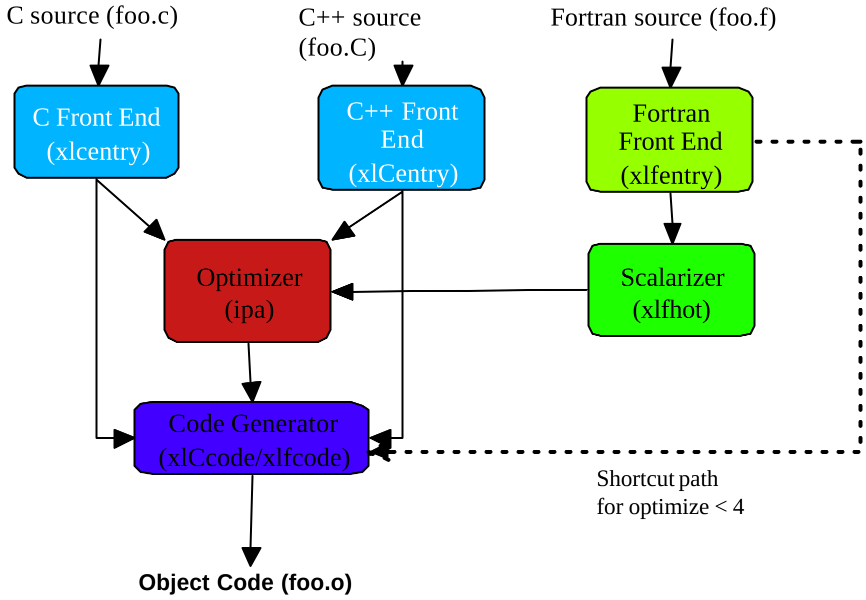
C for AIX version 5.0

- ANSI C89 compliant compiler (C99 coming soon)
- 32 and 64 bit support for serial and SMP
- Full support for OpenMP 1.0 (participating in OpenMP 2.0 definition)
- Support for TotalView, xldb, IBM distributed debugger and dbx/pdbx
- Snapshot directive for debugging optimized code
- Runtime memory debug support
- Portfolio of optimizing transformations
  - Similar to Fortran support but includes tuned optimizations for C pointers and systems coding styles
- More info: www.software.ibm.com/ad/caix

VisualAge for C++ for AIX version 5.0

- Fully compliant ANSI98 C++ compiler
- 32 and 64 bit support
- Batch compiler for traditional build environments and maximal optimization
- Incremental compiler for rapid application development (to be phased out in next release)
- Integrated graphical development environment including remote debug and performance visualization
- Support for TotalView, xldb, IBM distributed debugger and dbx/pdbx
- Portfolio of optimizing transformations
  - Subset of transformations available in Fortran and C but has tuned support for all processors
  - Much more coming soon
- More info: www.software.ibm.com/ad/vacpp

Inside a Compilation Step



All information subject to change without notice

Compiler Configuration Options

- **Configuration file (/etc/xlf.cfg, /etc/vac.cfg)**
  - Specifies default options and component and library paths for various compiler *invocations*
  - A compiler invocation (eg. xlf, xlf95, xlf_r) is simply a symbolic link to the compiler driver program (ie. argv[0]) used to obtain a specific behaviour - each has an associated stanza in the configuration file
  - You can specify your own configuration file using -F and you can create new stanzas ... take care if changing existing ones
- **-qpath option (old form uses -B and -t)**
  - Can be used to direct the compiler to use a component (eg. xlcentry, xlfcode) from a specific path
    - mixing components from different releases/PTFs works in general but is not warranted
    - recommend using a consistent set of components
  - Specifying a component other than the default one may require the use of a compatible runtime (ie. through the use of -L or LIBPATH) and a compatible message system (ie. through the use of NLSPATH)
- **TMPDIR**
  - Specifies an alternate directory for compiler temporary file storage - may be useful when using the -qipa option (including -O4, -O5)

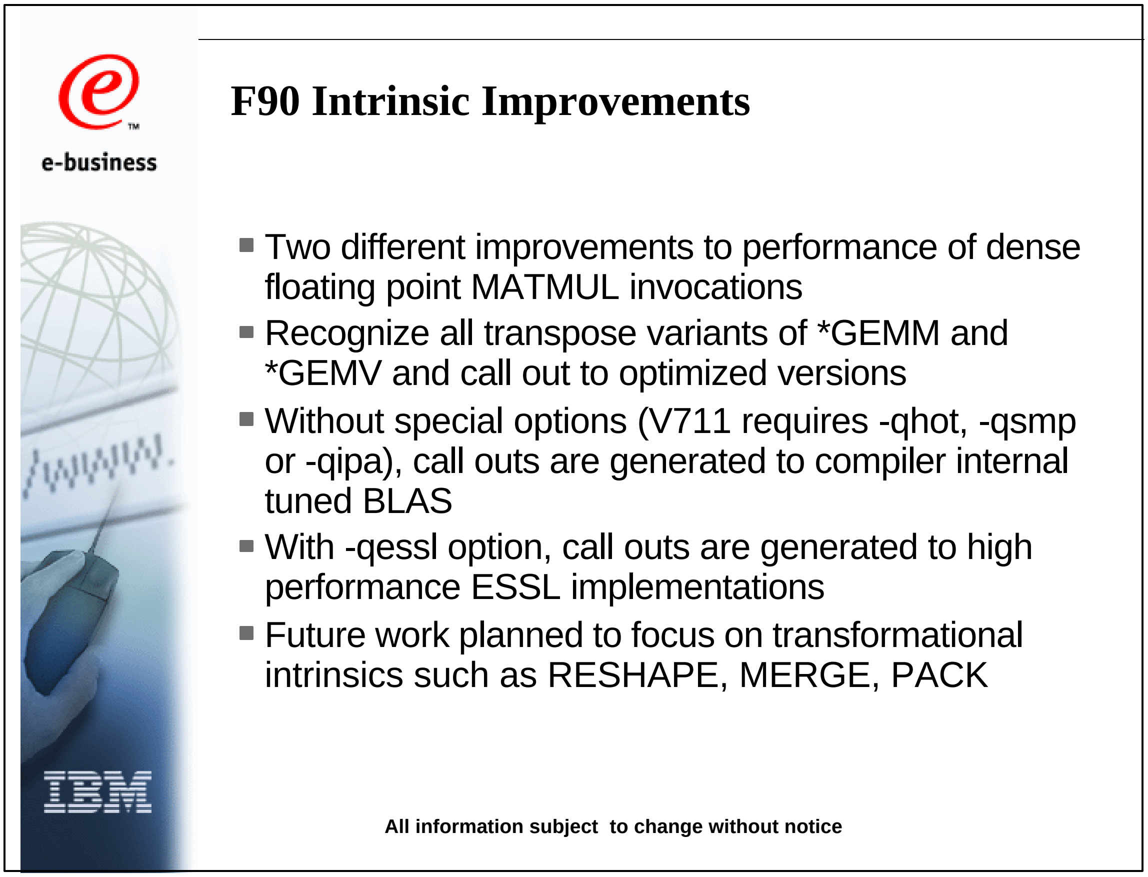
Configuration Options: Multiple Concurrent Compiler Versions

- XL Fortran manual documents a procedure for installing a compiler release in an alternate location
- A similar technique can be used for C/C++
- A technique based on alternate configuration files can be used to create a mirror image of an installed compiler in a specified location.
- We understand that these procedures are poorly documented and difficult to understand and manage
- We will better document and support these types of concurrent installations in XL Fortran V8 and VAC/C++ V6 with these caveats:
  - PMRs would be accepted only for compilers installed through smit/installp
  - PTFs or efixes would not apply to compilers installed in alternate locations

XL Fortran Roadmap

- **XL Fortran 7.1.1 - GA 12/01**
  - Power 4 optimization
  - Improved F90 intrinsic performance (MATMUL, TRANSPOSE)
  - Improved SMP performance (Auto and OpenMP)
  - Improved compile performance
- **XL Fortran 8.1 - projected GA 2Q02**
  - OpenMP 2.0
  - Fortran 200x subset
    - IEEE intrinsic module
    - allocatable components
  - Even more Power 4 optimization
  - Whole program analysis for locality and auto SMP parallelism (-qipa + -qhot or -qsmp=auto)

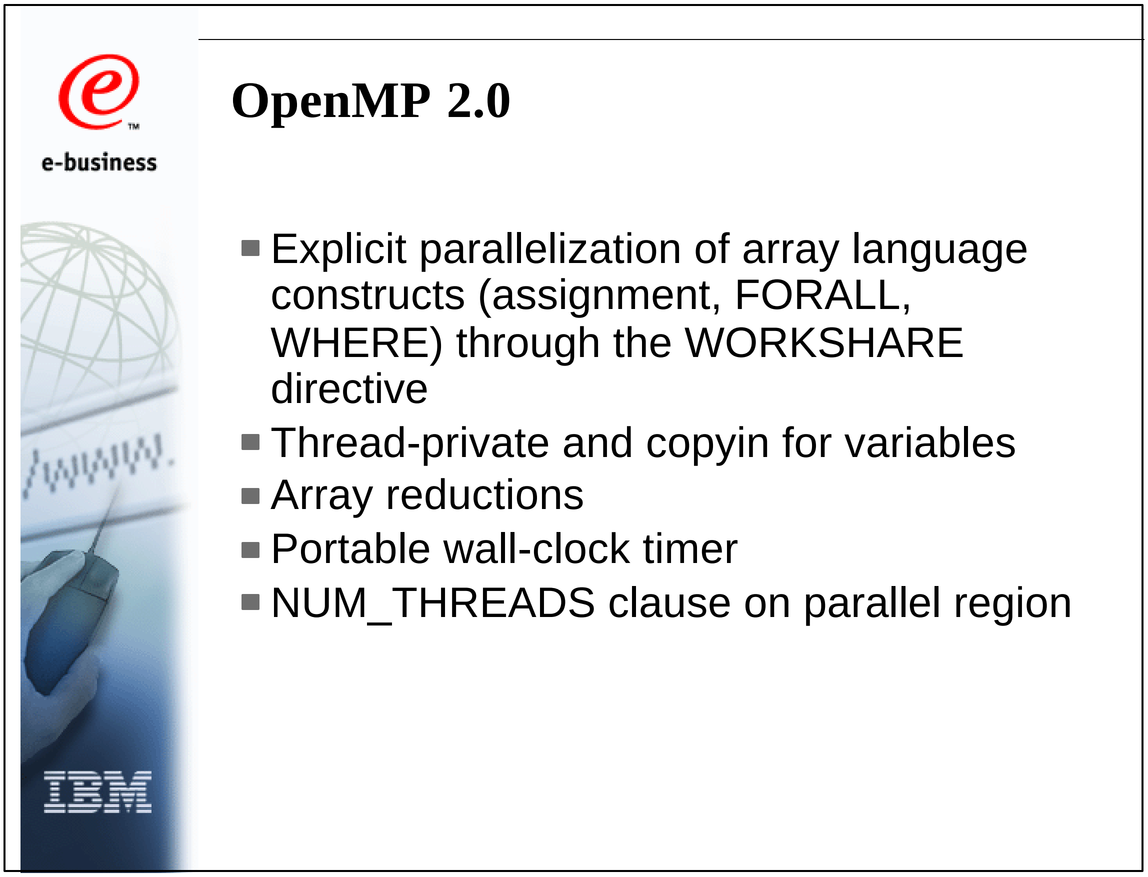
All information subject to change without notice



F90 Intrinsic Improvements

- Two different improvements to performance of dense floating point MATMUL invocations
- Recognize all transpose variants of *GEMM and *GEMV and call out to optimized versions
- Without special options (V711 requires -qhot, -qsmp or -qipa), call outs are generated to compiler internal tuned BLAS
- With -qessl option, call outs are generated to high performance ESSL implementations
- Future work planned to focus on transformational intrinsics such as RESHAPE, MERGE, PACK

All information subject to change without notice



OpenMP 2.0

- Explicit parallelization of array language constructs (assignment, FORALL, WHERE) through the WORKSHARE directive
- Thread-private and copyin for variables
- Array reductions
- Portable wall-clock timer
- NUM_THREADS clause on parallel region

C for AIX Roadmap

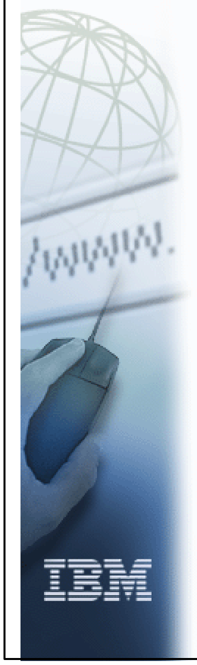
- **C for AIX 6.0 beta - GA 12/01**
  - Power 4 optimization
  - Improved SMP performance (Auto and OpenMP)
  - Optimization of loops for locality (-qhot)
- **C for AIX 6.0 - projected GA 2Q02**
  - ANSI C99 support, notably:
    - _Bool and _Complex type specifiers
    - signed and unsigned long long
    - restricted pointers
    - inline functions
    - variable size automatic arrays
  - Many GNU C compatible language and preprocessor extensions
  - Even more Power 4 optimization
  - Whole program analysis for locality and auto SMP parallelism (-qipa + -qhot or -qsmp=auto)

All information subject to change without notice

VisualAge for C++ for AIX Roadmap

- **VisualAge for C++ for AIX 6.0 beta - GA 12/01**
  - Power 4 optimization
  - Template code size optimizations
  - Optimization of virtual function calls
  - Improved optimization in the presence of exception handling
  - Whole program analysis (-qipa)
  - Optimization of loops for locality and auto SMP parallelism (-qhot, -qsmp=auto)
- **VisualAge for C++ for AIX 6.0 - projected GA 2Q02**
  - OpenMP 1.0 support (participating in OpenMP 2.0 definition)
  - Many GNU C/C++ compatible language and preprocessor extensions
  - Snapshot directive for debugging optimized code
  - Even more Power 4 optimization
  - Whole program analysis for locality and auto SMP parallelism (-qipa + -qhot or -qsmp=auto)

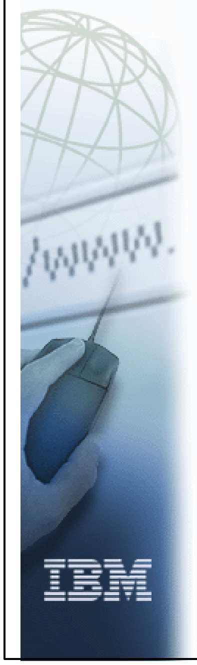
All information subject to change without notice



Common Optimization Technology Roadmap (Power 4 specific)

- **Architecture-neutral and -specific code paths**
  - ►tuning for arch=ppc and arch=pwr4
- **Precise machine model for scheduling (-O2+)**
  - ►new instruction scheduler with more detailed modelling capability
  - ►tuned through extensive experimention on early h/w
- **New loop transformations for deep pipelines (-O3+)**
  - ►more precise loop unrolling and pipelining
- **New aggressive branch optimizations (-O2+)**
  - ►branch pattern replacement
  - ►utilization of branch hints (eg. using profile feedback)
- **Optimized usage of hardware-expanded instructions**
  - ►eg. load/store update, mtcr, lm/stm
- **Optimized prefetch buffer allocation (-qhot)**
  - ►utilization of prefetch stream start instructions
  - ►loop nest fusion and partitioning to optimize # streams

All information subject to change without notice



Common Optimization Technology Roadmap

- **Enhanced profile-directed optimization (-qpdf)**
  - ► Profile-directed inlining, specialization, code motion, loop optimization
  - ► Much faster instrumentation (-qpdf1) time (20-40% penalty vs. 2-5x)
- **Interprocedural optimization (-qipa)**
  - ► Improved link time and reduced memory requirements
  - ► Better handling of multiple levels of pointer indirection
  - ► Better handling of function pointer calls, virtual calls, C++ exceptions, templates
  - ► Much expanded database of known library behaviours (eg. pthreads, MPI, ESSL, MASS)

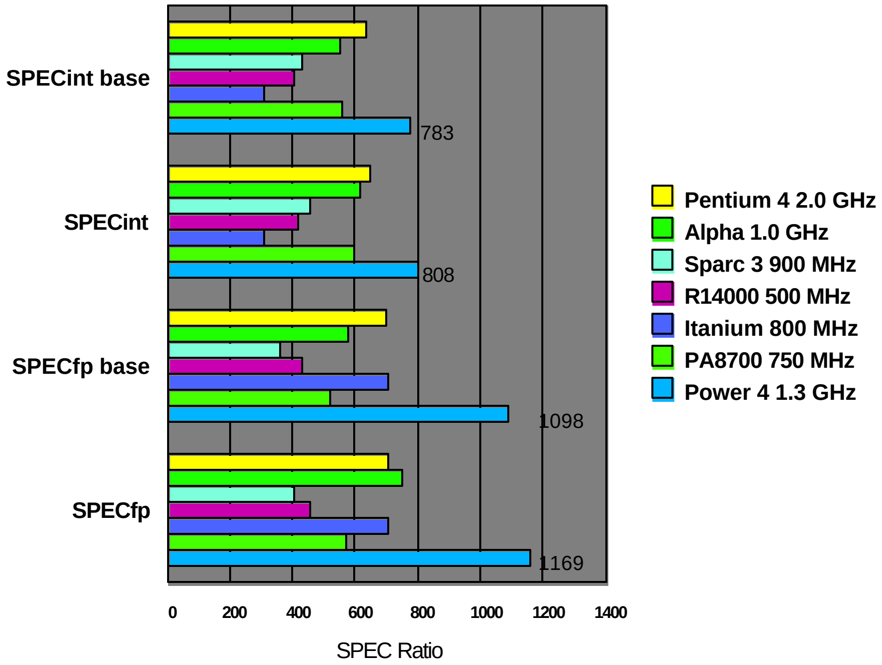
All information subject to change without notice

Common Optimization Technology Roadmap *(continued)*

- **Loop optimization (-qhot, -qsmp=auto)**
  - ►Improved compile time
  - ►More precise data dependence analysis leading to new opportunities for nest fusion, partitioning, interchange, parallelization, vectorization, etc
  - ►More aggressive transformations including improved handling of triangular and imperfect nests, indirect array indexing and branching within loops
- **Whole program optimization for locality and auto SMP parallelism (-qipa + -qhot or -qsmp=auto)**
  - ►Better handling of loop nests with calls
  - ►Inlining and cloning tuned to loop optimization (eg. fusion)
  - ►Improved handling of partial array read/write (eg. column), even through reference parameters
    - Leads to coarser grain (ie. more profitable) automatic parallelism

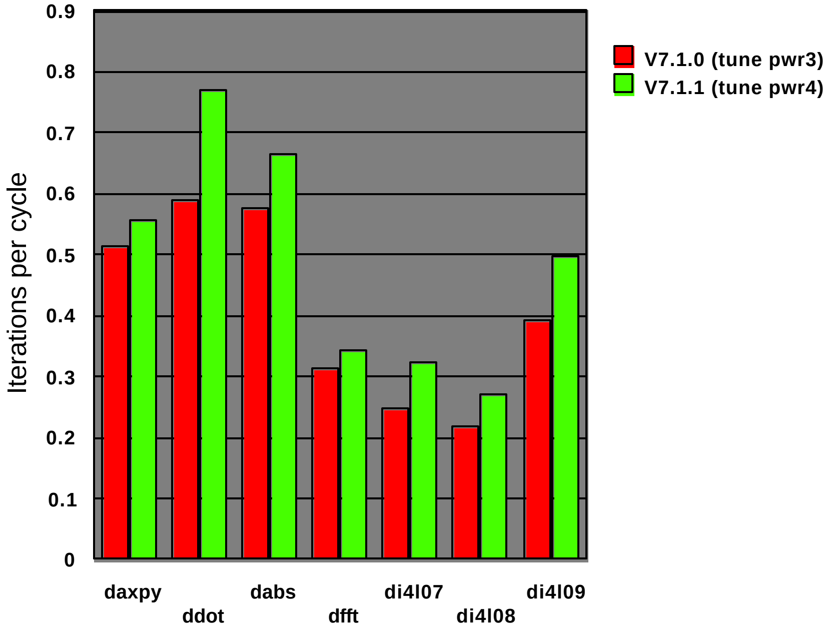
All information subject to change without notice

SPEC results for Power4

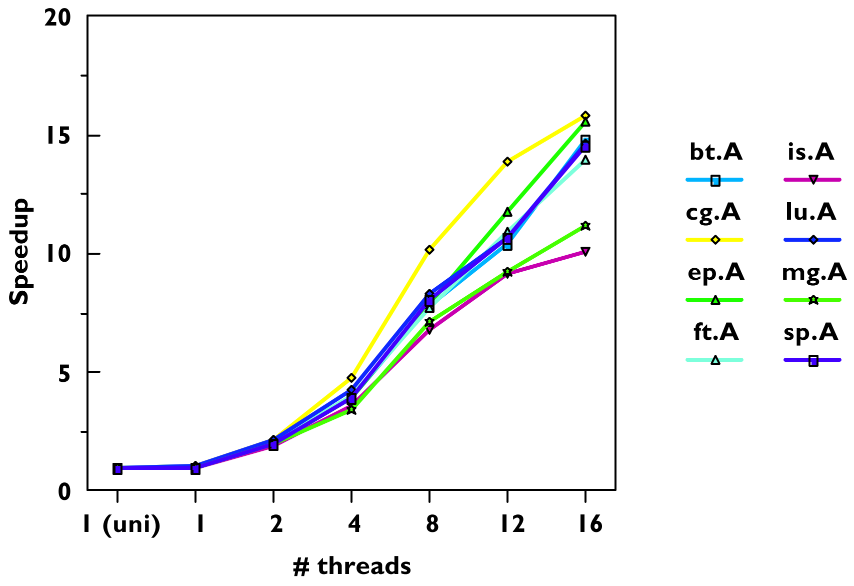


Competitive data from www.spec.org

Selected Kernel Performance

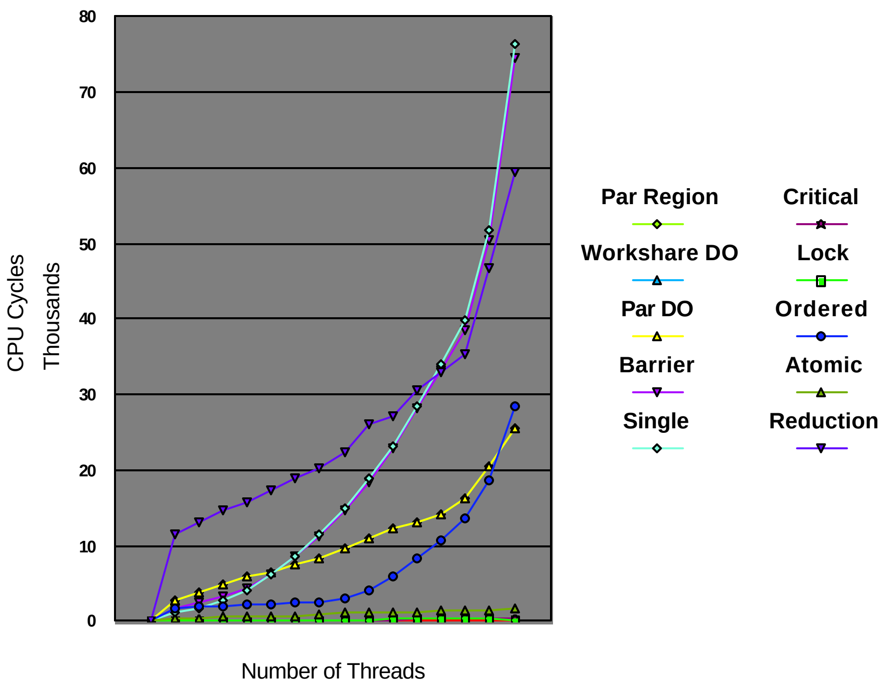


NAS OpenMP Speedup (16w NH-2)



e-business

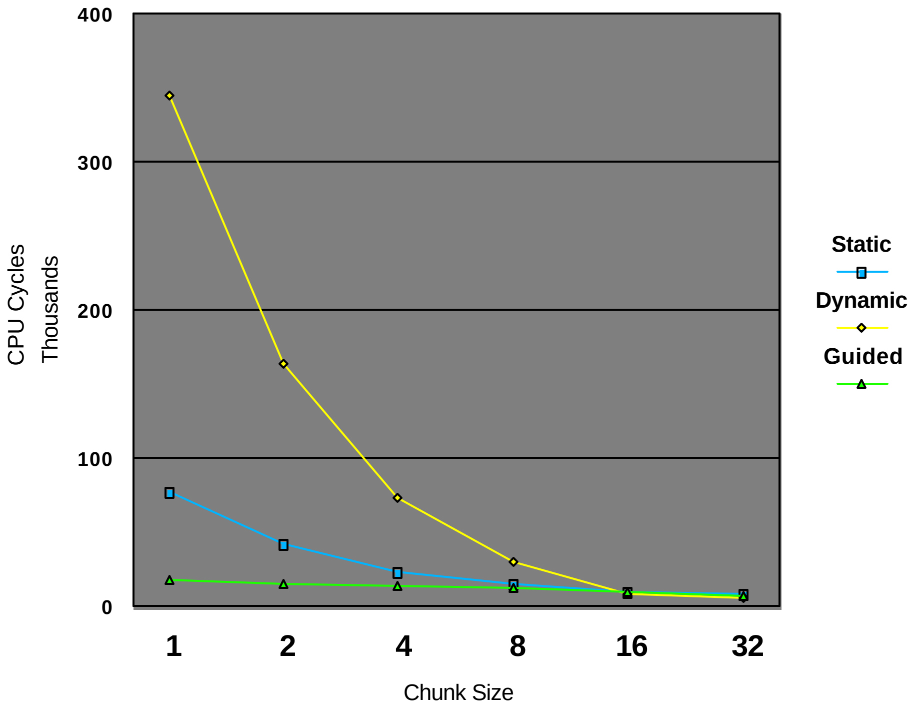
EPCC SyncBench Results - 16w NH-2



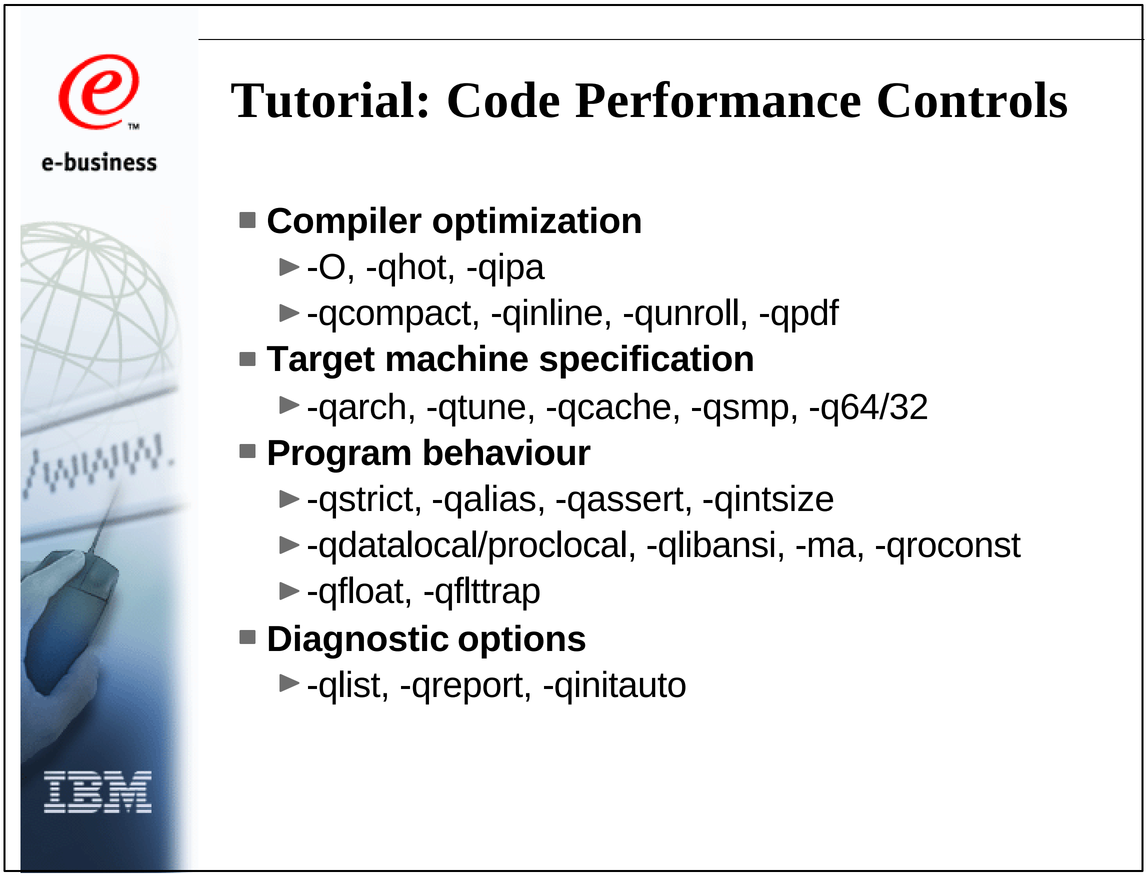
IBM

e-business

EPCC SchedBench Results - 16w NH-2

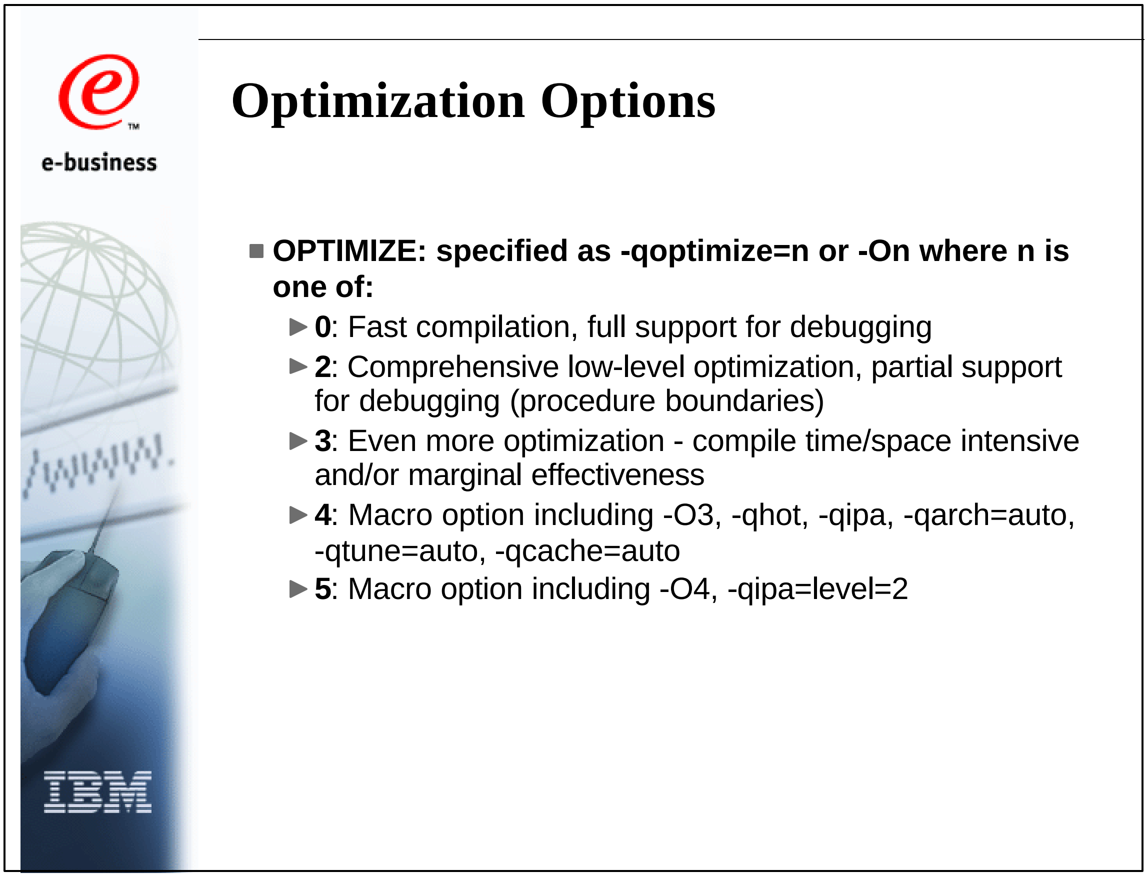


IBM



Tutorial: Code Performance Controls

- **Compiler optimization**
  - -O, -qhot, -qipa
  - -qcompact, -qinline, -qunroll, -qpdf
- **Target machine specification**
  - -qarch, -qtune, -qcache, -qsmp, -q64/32
- **Program behaviour**
  - -qstrict, -qalias, -qassert, -qintsize
  - -qdatalocal/proclocal, -qlibansi, -ma, -qroconst
  - -qfloat, -qflttrap
- **Diagnostic options**
  - -qlist, -qreport, -qinitauto



Optimization Options

- **OPTIMIZE: specified as -qoptimize=n or -On where n is one of:**
  - **0**: Fast compilation, full support for debugging
  - **2**: Comprehensive low-level optimization, partial support for debugging (procedure boundaries)
  - **3**: Even more optimization - compile time/space intensive and/or marginal effectiveness
  - **4**: Macro option including -O3, -qhot, -qipa, -qarch=auto, -qtune=auto, -qcache=auto
  - **5**: Macro option including -O4, -qipa=level=2

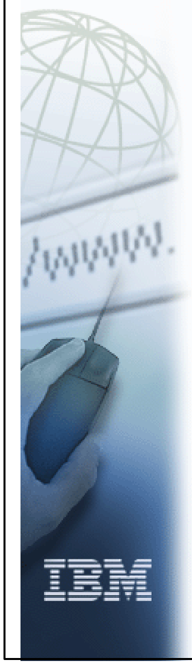
Optimization Options (*continued*)

- **HOT (High Order Transformations) - Fortran (C and C++ coming soon)**
  - Specified as -qhot[=[no]vector | arraypad[=*n*]]
  - Optimized handling of F90 array language constructs (elimination of temporaries, fusion of statements)
  - High level transformation (eg. interchange) of loop nests to improve memory locality (reduce cache/TLB misses), optimize usage of hardware prefetch and balance loop computation (typically ld/st vs. float)
  - *Optionally* transforms loops to exploit vector intrinsic library (eg. reciprocal, sqrt, trig) - may result in slightly different rounding
  - *Optionally* introduces array padding under user control - potentially unsafe if not applied uniformly

Optimization Options (*continued*)

- **IPA (Inter-Procedural Analysis) - Fortran and C (C++ coming soon)**
  - Specified as -qipa[=level=*n* | inline= | *fine tuning*] on both compile *and* link steps
  - Expand the scope of optimization to an entire program unit (executable or shared object)
  - *level=0*: Program partitioning and simple interprocedural optimization
  - *level=1*: Inlining and global data mapping
  - *level=2*: Global alias analysis, specialization, interprocedural data flow
  - *inline=*: Precise user control of inlining
  - *fine tuning*: Specify library code behaviour, tune program partitioning, read commands from a file

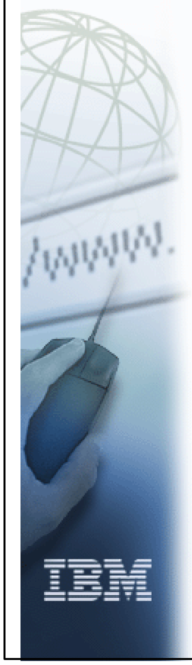
e-business



Target Machine Options

- **ARCH**
  - Restricts the compiler to generate a subset of the Power or PowerPC instruction set
  - Specified as -qarch=*isa* where *isa* is one of:
    - *com* (default): Code can run on any RS/6000 - implies -qtune=pwr2
    - *auto*: Code may take advantage of instructions available only on the compiling machine (or similar machines)
    - *ppc*: Code follows PowerPC architecture - implies -qtune=604 (32 bit) or -qtune=pwr3 (64 bit)
    - *pwr3*: Code can run on any Power 3 - implies -qtune=pwr3
    - Lots of others: *pwr, pwr2, 604, ...*

e-business



Target Machine Options *(continued)*

- **TUNE:** Bias optimization toward execution on a given machine
  - Does *not* imply anything about the ability to run correctly on a given machine - only affects performance
  - Specified as -qtune=*machine* where *machine* is one of auto, 604, pwr2, p2sc, pwr3, rs64a, etc.
- **CACHE**: Defines a specific cache/memory geometry
  - Defaults are set through TUNE
  - Specified as -qcache=level=*n*:*cache_spec*, where *cache_spec* includes:
    - type=i|d|c: cache type (instruction/data/combined)
    - line=*lsz*:size=*sz*:assoc=*as*: line/cache size and set associativity
    - cost=*c*: cost (in cpu cycles) of a miss

Target Machine Options *(continued)*

- **64/32**: Generate code for 64 bit (4/8/8) or 32 bit (4/4/4) addressing model
  - ► Specified as -q32 or -q64
- **SMP (Fortran, C)**: Generate threaded code for a shared-memory parallel machine
  - ► Specified as -qsmp[=[no]auto:=[no]omp:*fine tuning*]
  - ► *auto* instructs the compiler to automatically generate parallel code where possible without user assistance
  - ► *omp* instructs the compiler to observe OpenMP 1.0 language extensions for specifying explicit parallelism
  - ► *fine tuning* includes control over thread scheduling, nested parallelism and locking

Program Behaviour Options

- **STRICT**
  - ► Specified as -q[no]strict, default is -qstrict with -qoptimize=0 and -qoptimize=2, -qnostrict with -qoptimize=3,4,5
  - ► *nostrict* allows the compiler to reorder floating point calculations and potentially excepting instructions
- **ALIAS (Fortran)**
  - ► Specified as -qalias=[no]std:[no]aryovrlp:*others*
  - ► Allows the compiler to assume that certain variables do not refer to overlapping storage
  - ► *std* (default) refers to the rule about storage association of reference parameters with each other and globals
  - ► *aryovrlp* (default) defines whether there are any assignments between storage-associated arrays - try -qalias=noaryovrlp for better performance

Program Behaviour Options (*continued*)

- **ALIAS (C, C++)**
  - Similar to Fortran option of the same name but focussed on overlap of storage accessed using pointers
  - Specified as -qalias=*subopt* where *subopt* is one of:
    - *[no]ansi*: Enable ANSI standard type-based alias rules
    - *[no]typeptr*: Assume pointers to different types never point to the same or overlapping storage
    - *[no]allptrs*: Assume that different pointer variables always point to non-overlapping storage
    - *[no]addrtaken*: Assume that external variables do not have their address taken outside the source file being compiled

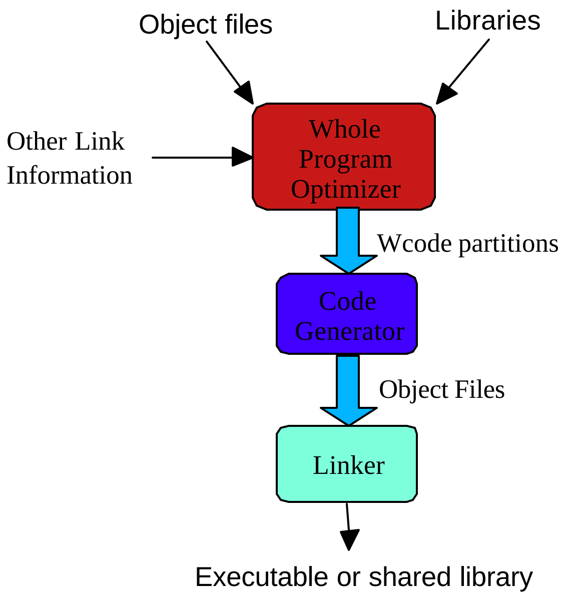
Directives and Pragmas

- **OpenMP 1.0** - supported in C and Fortran
- **Legacy SMP** directives and pragmas
  - Most of these are superceded by OpenMP - use OpenMP where possible
- **Assertive directives** (Fortran)
  - ASSERT, INDEPENDENT, CNCALL, PERMUTATION
- **Assertive pragmas** (C)
  - *isolated_call*, *disjoint, independent_loop, independent_calls, iterations, permutation, execution_frequency, leaves*
- **Embedded Options**
  - *#pragma options* and *#pragma option_override* in C
  - @PROCESS in Fortran
- **Prescriptive directives** (Fortran)
  - PREFETCH, UNROLL
- **Prescriptive pragmas** (C)
  - *sequential_loop*

Diagnostic Options

- **LIST**
  - Specified as -qlist
  - Instructs the compiler to emit an object listing
  - The object listing includes hex and pseudo-assembly representations of the generated code along with traceback tables and text constants
- **REPORT (Fortran)**
  - Specified as -qreport [=smplist]
  - Instructs the high level optimizer to emit a report including pseudo-Fortran along with annotations describing what transformations were performed (eg. loop unrolling, automatic parallelization)
  - Also includes information about data dependences and other inhibitors to optimization

Inside an Link-time Compilation

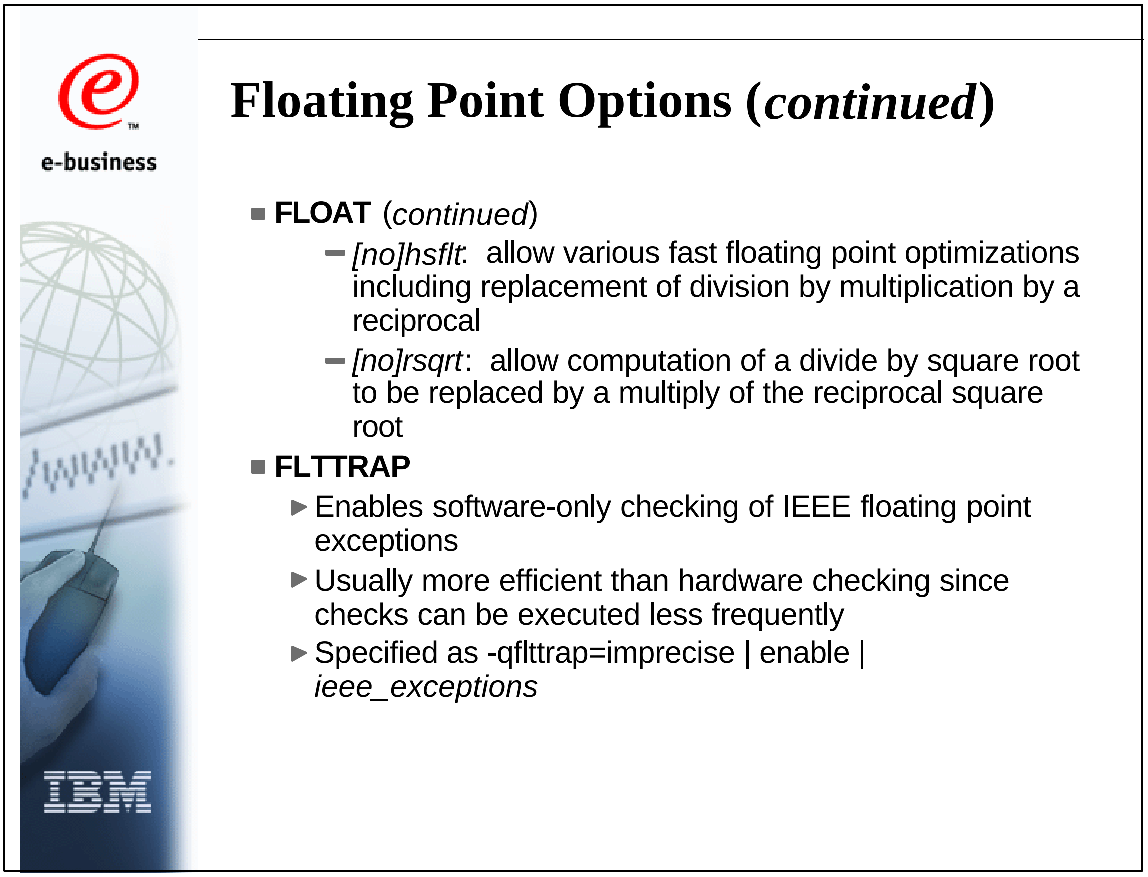


All information subject to change without notice

Backup

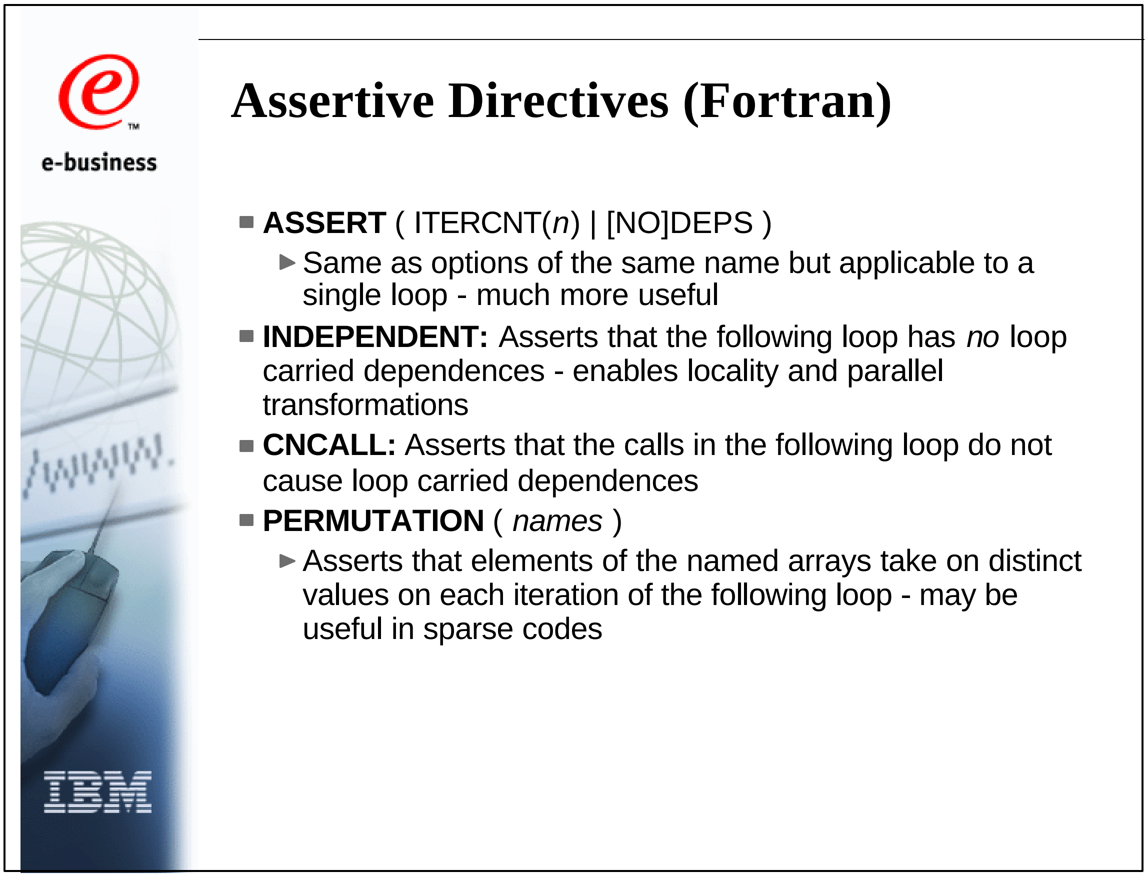
Floating Point Options

- **FLOAT**
  - Precise control over the handling of floating point calculations
  - Specified as -qfloat=*subopt* where *subopt* is one of:
    - *[no]fold*: enable compile time evaluation of floating point calculations - may want to disable for handling of certain exceptions (eg. overflow, imprecise)
    - *[no]maf*: enable generation of multiple-add type instructions - may want to disable for exact compatibility with other machines but this will come at a high price in performance
    - *[no]rrm*: specifies that rounding mode may not be round-to-nearest (default is *norrm*)



Floating Point Options (*continued*)

- **FLOAT** (*continued*)
  - *[no]hsflt*: allow various fast floating point optimizations including replacement of division by multiplication by a reciprocal
  - *[no]rsqrt*: allow computation of a divide by square root to be replaced by a multiply of the reciprocal square root
- **FLTTRAP**
  - Enables software-only checking of IEEE floating point exceptions
  - Usually more efficient than hardware checking since checks can be executed less frequently
  - Specified as -qflttrap=imprecise | enable | *ieee_exceptions*



Assertive Directives (Fortran)

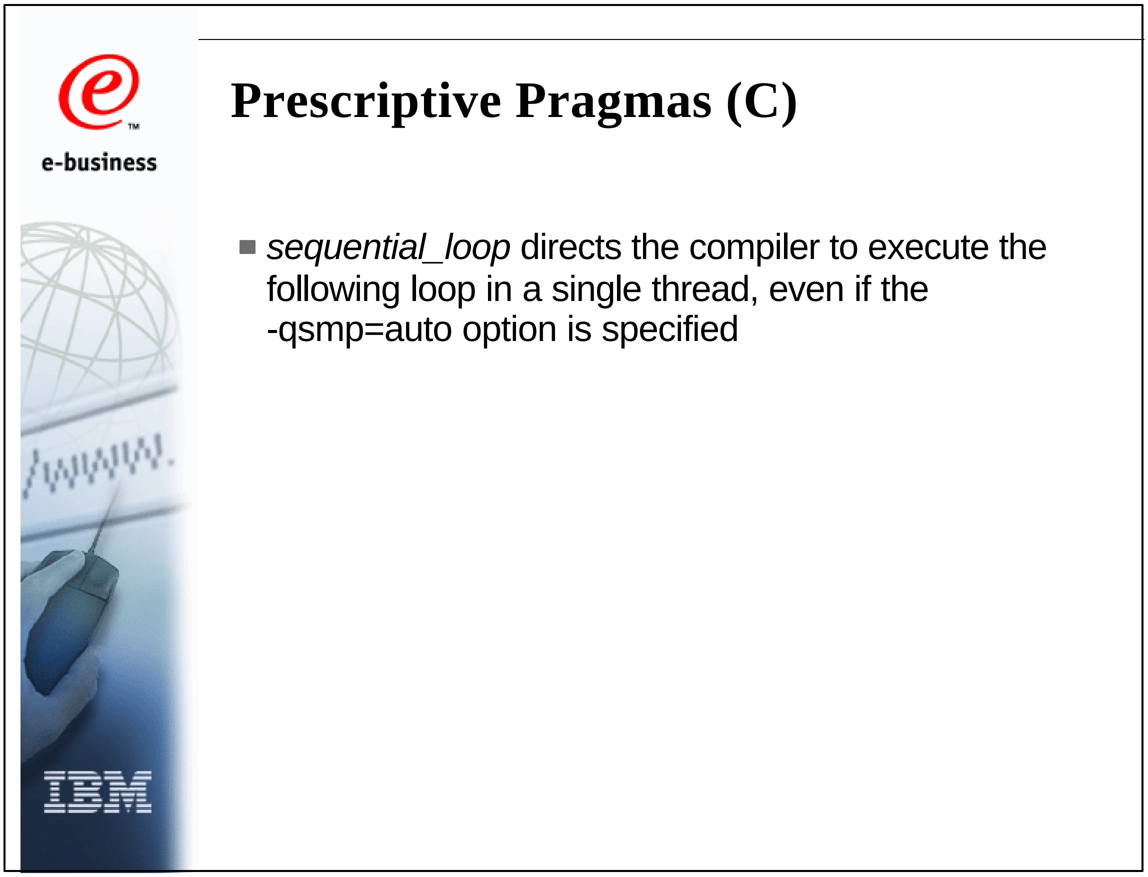
- **ASSERT** (ITERCNT(*n*) | [NO]DEPS)
  - Same as options of the same name but applicable to a single loop - much more useful
- **INDEPENDENT:** Asserts that the following loop has *no* loop carried dependences - enables locality and parallel transformations
- **CNCALL:** Asserts that the calls in the following loop do not cause loop carried dependences
- **PERMUTATION** (*names*)
  - Asserts that elements of the named arrays take on distinct values on each iteration of the following loop - may be useful in sparse codes

Assertive Pragmas (C)

- *isolated_call* (*function_list*) asserts that calls to the named functions do not have side effects
- *disjoint* (*variable_list*) asserts that none of the named variables share overlapping areas of storage
- *independent_loop* is equivalent to INDEPENDENT
- *independent_calls* is equivalent to CNCALL
- *permutation* is equivalent to PERMUTATION
- *iterations* is equivalent to ASSERT(ITERCNT)
- *execution_frequency* (*very_low*) asserts that the control path containing the pragma will be infrequently executed
- *leaves* (*function_list*) asserts that calls to the named functions will not return (eg. exit)

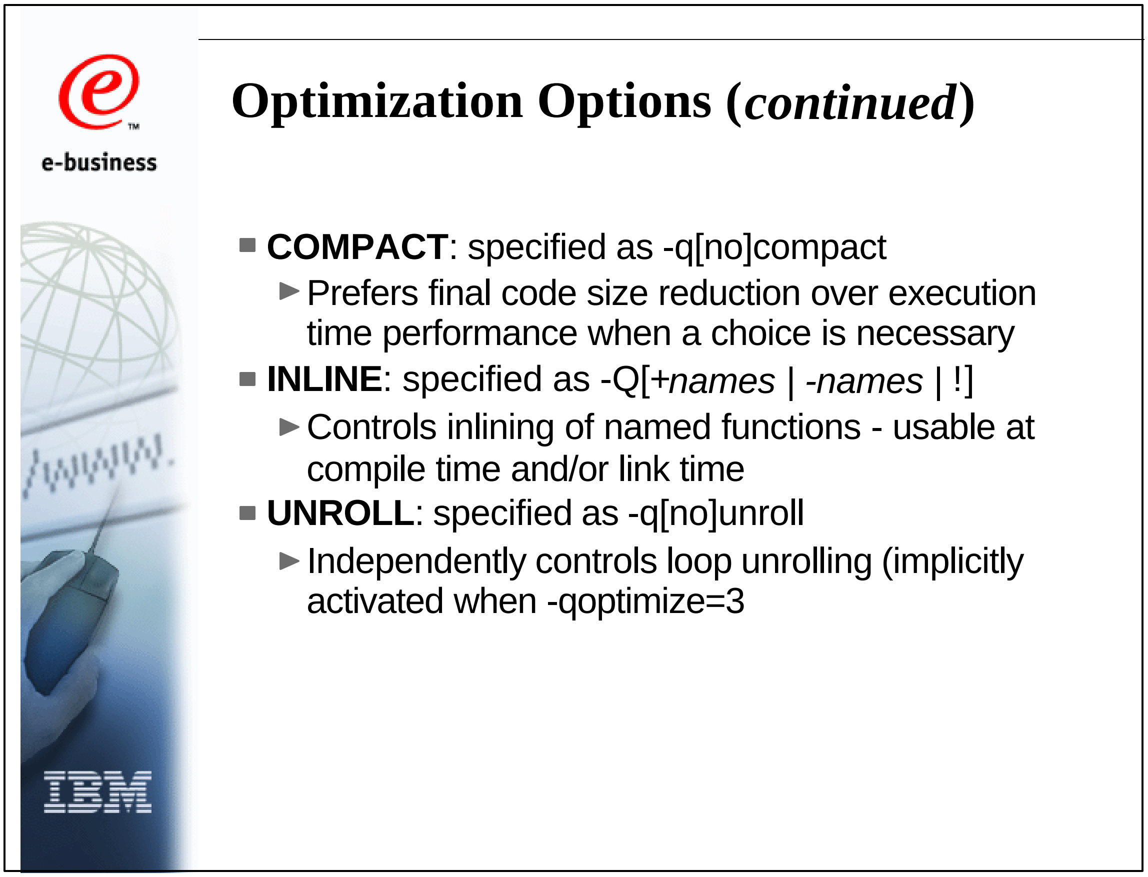
Prescriptive Directives (Fortran)

- **PREFETCH**
  - PREFETCH_BY_LOAD (*variable_list*): issue *dummy* loads to cause the given variables to be prefetched into cache - useful on Power machines or to activate Power 3 hardware prefetch
  - PREFETCH_FOR_LOAD (*variable_list*): issue a *dcbt* instruction for each of the given variables.
  - PREFETCH_FOR_STORE (*variable_list*): issue a *dcbtst* instruction for each of the given variables.
- **UNROLL**
  - Specified as [NO]UNROLL [(*n*)]
  - Used to activate/deactivate compiler unrolling for the following loop.
  - Can be used to give a specific unroll factor.



Prescriptive Pragmas (C)

- *sequential_loop* directs the compiler to execute the following loop in a single thread, even if the -qsmp=auto option is specified



Optimization Options (*continued*)

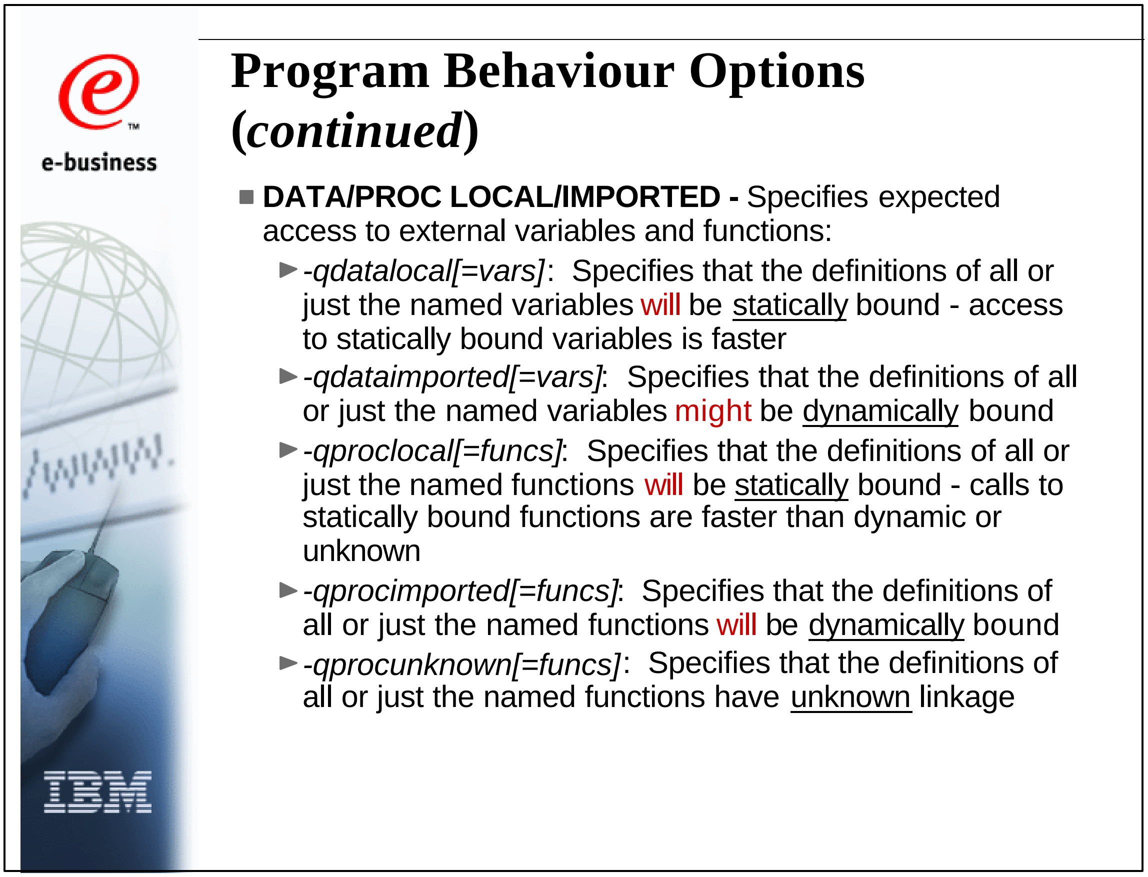
- **COMPACT**: specified as -q[no]compact
  - Prefers final code size reduction over execution time performance when a choice is necessary
- **INLINE**: specified as -Q[+*names* | -*names* | !]
  - Controls inlining of named functions - usable at compile time and/or link time
- **UNROLL**: specified as -q[no]unroll
  - Independently controls loop unrolling (implicitly activated when -qoptimize=3

Optimization Options (*continued*)

- **INLGLUE** - Specified as *-q[no]inlglue*
  - Inline calls to "glue" code used in calls through function pointers (including *virtual*) and calls to functions which are dynamically bound
- **TBTABLE**
  - Controls the generation of traceback table information:
  - *-qtbtable=none* inhibits generation of tables - no stack unwinding is possible
  - *-qtbtable=small* generates tables which allow stack unwinding but omit name and parameter information - useful for optimized C++
  - *-qtbtable=full* generates full tables including name and parameter information - useful for debugging

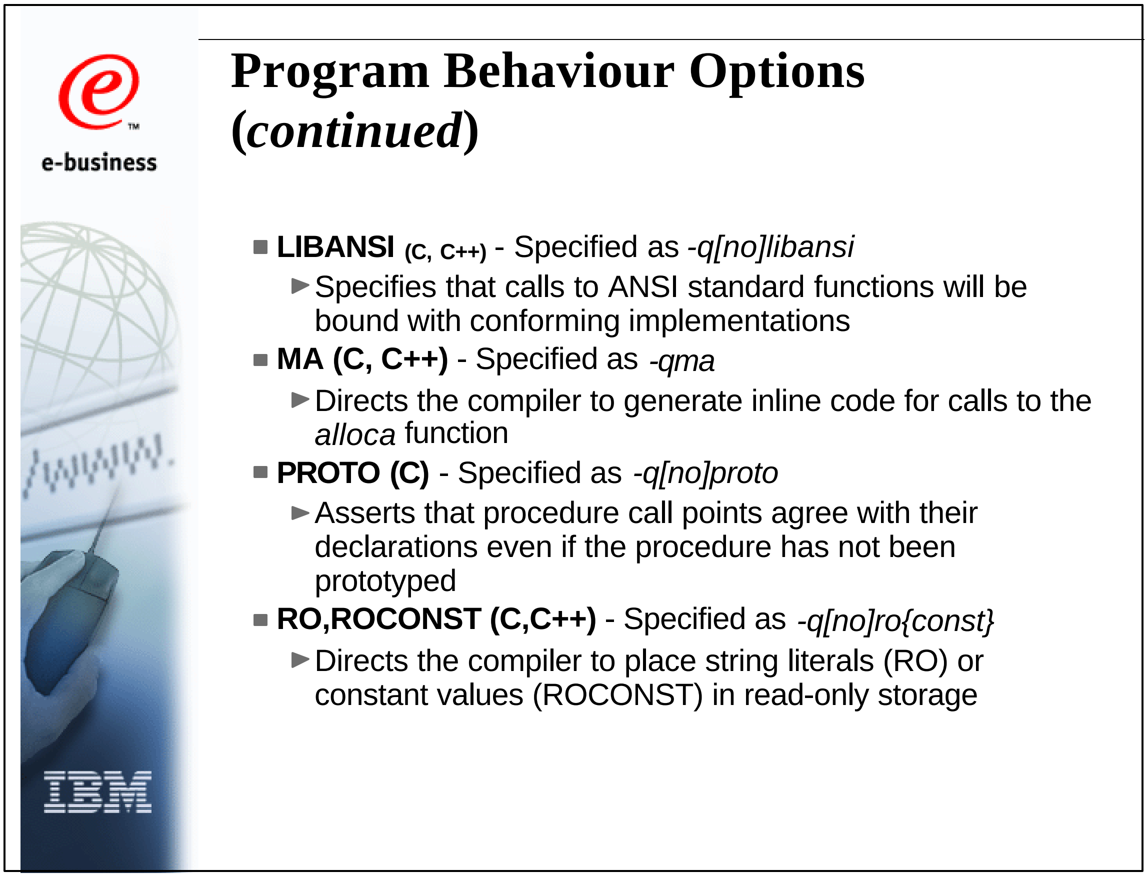
Program Behaviour Options (*continued*)

- **ASSERT (Fortran, C)**
  - Specified as -qassert=[no]deps | itercnt=*n*
  - *deps* (default) indicates that some loop has a loop carried memory dependence - try -qassert=nodeps for improved performance
  - *itercnt* modifies the default assumptions about the expected iteration count of loops (normally 10)
- **INTSIZE (Fortran)**: Define the default size of INTEGER variables
  - Specified as -qintsize=1|2|4|8
  - When using -q64, try -qintsize=8 for improved performance
- **IGNERRNO (C,C++)** - Specified as *-q[no]ignerrno*
  - Indicates that the value of *errno* is not needed by the program



Program Behaviour Options (*continued*)

- **DATA/PROC LOCAL/IMPORTED -** Specifies expected access to external variables and functions:
  - ► *-qdatalocal[=vars]*: Specifies that the definitions of all or just the named variables will be statically bound - access to statically bound variables is faster
  - ► *-qdataimported[=vars]*: Specifies that the definitions of all or just the named variables might be dynamically bound
  - ► *-qproclocal[=funcs]*: Specifies that the definitions of all or just the named functions will be statically bound - calls to statically bound functions are faster than dynamic or unknown
  - ► *-qprocimported[=funcs]*: Specifies that the definitions of all or just the named functions will be dynamically bound
  - ► *-qprocunknown[=funcs]*: Specifies that the definitions of all or just the named functions have unknown linkage



Program Behaviour Options (*continued*)

- **LIBANSI** **(C, C++)** - Specified as *-q[no]libansi*
  - ► Specifies that calls to ANSI standard functions will be bound with conforming implementations
- **MA (C, C++)** - Specified as *-qma*
  - ► Directs the compiler to generate inline code for calls to the *alloca* function
- **PROTO (C)** - Specified as *-q[no]proto*
  - ► Asserts that procedure call points agree with their declarations even if the procedure has not been prototyped
- **RO,ROCONST (C,C++)** - Specified as *-q[no]ro{const}*
  - ► Directs the compiler to place string literals (RO) or constant values (ROCONST) in read-only storage